**Министерство науки и высшего образования Российской Федерации**

**федеральное государственное бюджетное образовательное учреждение высшего образования**

**Московский авиационный институт**

**(Национальный исследовательский университет)**

Факультет №3

«Системы управления, информатика и электроэнергетика»

Кафедра 304

**Отчет по лабораторной работе**

по учебной дисциплине «Программирование»

на тему «Алгоритмы поиска»

Руководители работы

\_\_\_\_\_\_\_\_\_\_\_\_Чечиков Ю. Б.

«\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_2022 г.

\_\_\_\_\_\_\_\_\_\_\_\_ Секретарев В. Е.

«\_\_\_\_»\_\_\_\_\_\_\_\_\_\_ 2022 г.

Исполнители

обучающиеся группы М30-210Б-21

\_\_\_\_\_\_\_\_\_\_\_\_ Папуашвили Г. Д.

«\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_ 2022 г.

\_\_\_\_\_\_\_\_\_\_\_\_ Шилин И. С.

«\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_ 2022 г.

Оглавление

[Задание 3](#_Toc117080937)

[Структурные схемы алгоритма 4](#_Toc117080938)

[Код программы 10](#_Toc117080939)

[Диаграммы 18](#_Toc117080940)

[Вывод 19](#_Toc117080941)

# Задание

**Цель работы**: изучить основные принципы работы алгоритмов поиска, исследовать их свойства:

* алгоритм **BLS** - последовательный поиск (Better\_Linear\_Search);
* алгоритм **SLS** - быстрый последовательный поиск (Sentinel\_Linear);
* алгоритм **Т** - последовательный поиск в упорядоченном массиве, в конце массива помещается фиктивная запись, значение которой намного больше значения ключа поиска;
* алгоритм **В** - бинарный поиск.

**Задание**

Для алгоритмов **BLS** и **SLS** в качестве входного массива использовать одну и ту же последовательность значений (функция **rand**( )).

Для алгоритмов **Т** и **В** – значения массива должны быть отсортированы по неубыванию, одна и та же последовательность чисел (можно использовать соответствующую функцию из первой лабораторной работы).

Оценить длительность поиска для различных значений размеров последовательностей (начиная с 10000 до 200000 элементов массива, провести измерения не менее, чем для 10 разных размерностей).

Для каждой размерности рассматриваются случаи нахождения ключа поиска в начале, в середине и в конце массива.

Для алгоритмов **BLS** и **SLS** кроме подсчета **времени,** необходимого для поиска, требуется определить сколько раз выполняются операции **сравнения** (сравнение ключа с элементом массива, а также в одном из этих двух алгоритмов добавляется подсчет сравнений при анализе индекса элемента массива в цикле… ).

Все результаты оформить в виде таблиц и графиков. На графиках **- только временные характеристики** поиска.

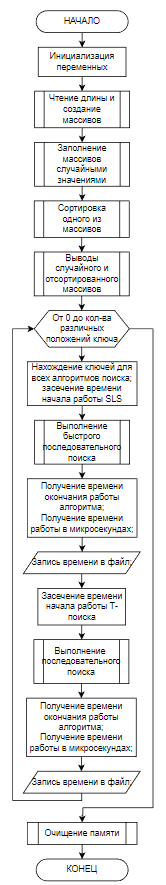
По результатам сделать выводы об эффективности того или иного алгоритма поиска.

**Отчет по лабораторной работе должен содержать:**

* задание;
* тексты программ;
* результаты работы программ и результаты сопоставительного анализа – в виде таблиц, графиков;
* выводы по работе.

# Структурные схемы алгоритма

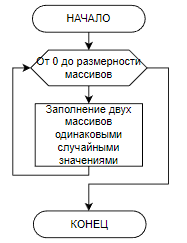
Модуль main



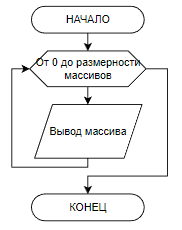
Модуль чтения и создания



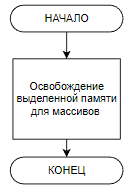
Модуль заполнения массива



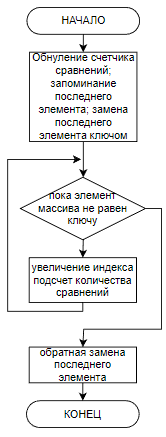
Модуль вывода массива



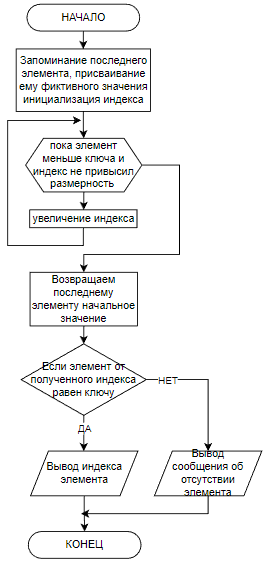
Модуль удаления



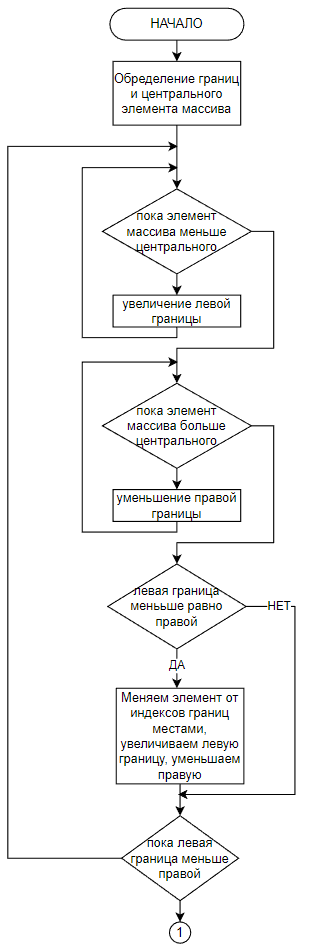
Модуль SLS

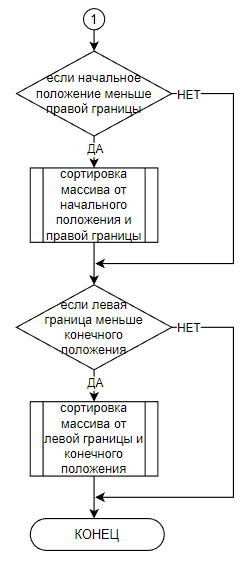


Модуль Т поиска

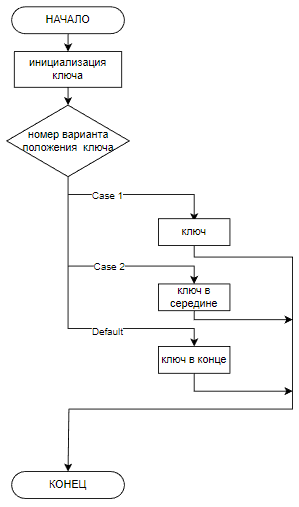


Модуль сортировки





Модуль смены ключа



# Код программы

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Курс ПРОГРАММИРОВАНИЕ \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Project type :Win64 Console Application \*

\* Project name :Lab2 \*

\* File name :Lab2.exe \*

\* Language :CPP14, CLION \*

\* Programmers :Папуашвили Георгий Давидович Шилини Иван Алексеевич, М3О-210Б-21 \*

\* Modified By : \*

\* Created :18.10.2022 \*

\* Last revision:19.10.2022 \*

\*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\*

\* Comment: Лабораторная работа №8 (лабораторная работа №2, 2-й курс) \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#include "functions.h"

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* ГЛОБАЛЬНЫЕ ПЕРЕМЕННЫЕ \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

const char \*FILE\_INPUT = "FILE\_INPUT.txt"; // Файл со входными данными

const char \*FILE\_OUTPUT[] = {"Time\_Comparisons.csv"}; // Файл с информацией о времени и кол-ве сравнений

ofstream stream\_out(\*FILE\_OUTPUT); // Открываем новый поток вывода с выходным файлом

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* ОСНОВНАЯ ПРОГРАММА \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

int main() {

setlocale(LC\_ALL, "Russian"); // Подключаем локализацию кириллицы

int \*array; // Указатель на несортированный массив

int \*arraySorted; // Указатель на сортированный массив

int size; // Инициализация длины массива

readAndCreate(&array, &arraySorted, size, &\*FILE\_INPUT); // Вызов функции чтения файла

fillingOfArrays(&array, &arraySorted, size); // Вызов функции наполнения массивов

cout << "-------rand--------\n";

arraysOutput(&array, size); // Вызов функции вывода массива в консоль

int leftBound = 0; // Задание левой границы

int rightBound = size - 1; // Задание правой границы

quickRecursiveSort(&arraySorted, leftBound, rightBound); // Вызов функции быстрой рекурсивной сортировки

cout << "-------rand sort--------\n";

arraysOutput(&arraySorted, size); // Вызов функции сортировки массива

cout << "=============================main part===========================================\n";

// Цикл смены позиции "ключа"

for (int i = 0; i < 3; i++) {

int keyOfSLS = changeKey(&array, i, size); // Ключ для смены индекса для SLS

int keyOfT = changeKey(&arraySorted, i, size); // Ключ для смены индекса для T\_find

//SLS

cout << "SLS" << ";;" << endl;

auto begin1 = chrono::steady\_clock::now(); // Установка таймера и начало отсчета

sentinelLinearSearch(&array, size, keyOfSLS); // Вызов функции SLS

auto end1 = std::chrono::steady\_clock::now(); // Остановка таймера

auto elapsed\_ms1 = std::chrono::duration\_cast<std::chrono::microseconds>(end1 - begin1); // Получение времени

cout << "time(ms)= ";

stream\_out << elapsed\_ms1.count() << ";;" << endl; // Запись времени в файл

cout << elapsed\_ms1.count() << ";" << endl;

cout << "-----------------------------------------\n";

//T\_FIND

cout << "T-find" << ";;" << endl;

auto begin2 = chrono::steady\_clock::now(); // Установка таймера и начало отсчета

T\_find(&arraySorted, size, keyOfT); // Вызов функции T\_find

auto end2 = std::chrono::steady\_clock::now(); // Остановка таймера

auto elapsed\_ms2 = std::chrono::duration\_cast<std::chrono::microseconds>(end2 - begin2); // Получение времени

cout << "time(ms)= ";

stream\_out << elapsed\_ms2.count() << ";;" << endl; // Запись времени в файл

cout << elapsed\_ms2.count() << ";" << endl;

cout << "-----------------------------------------\n";

}//for

deleteArrs(array, arraySorted); // Очищение памяти

return 0;

}

**HEADER**

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
\* ПРОТОТИПЫ ФУНКЦИЙ \*  
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  
  
#include <iostream> // Подключение библиотеки стандартных команд  
#include <cstdlib> // Подключение библиотеки  
#include <fstream> // Подключение библиотеки ввода/вывода  
#include <chrono> // Подключение библиотеки для таймера  
  
using namespace std; // Пространство имен std  
  
// Функция считывания и создания массивов  
void readAndCreate(  
 int \*\*array, // Динамический массив указателей  
 int \*\*arrayFutureSourted, // Отсортированный массив указателей  
 int &size, // Длина массива  
 const char \*FILE\_INPUT // Имя выходного файла  
);  
  
// Функция наполнения массивов  
void fillingOfArrays(  
 int \*\*array, // Динамический массив указателей  
 int \*\*arrayFutureSourted, // Отсортированный массив указателей  
 int size // Длина массива  
);  
  
// Функция вывода массива  
void arraysOutput(  
 int \*\*array, // Динамический массив указателей  
 int size // Длина массива  
);  
  
// Функция быстрой рекурсивной сортировки  
void quickRecursiveSort(  
 int \*\*array, // Динамический массив указателей  
 int left, // Левая граница массива  
 int right // Правая граница массива  
);  
  
// Функция линейного поиска  
void sentinelLinearSearch(  
 int \*\*array, // Динамический массив указателей  
 int SIZE, // Длина массива  
 int key // Ключ индексов  
);  
  
// Функция смена ключа  
int changeKey(  
 int \*\*Array, // Динамический массив указателей  
 int number, // Переменная для передачи значения ключа  
 int SIZE // Ключ индексов  
);  
  
// Функция поиска  
void T\_find(  
 int \*\*array, // Динамический массив указателей  
 int SIZE, // Длина массива  
 int key // Ключ индексов  
);  
  
// Функция очищения памяти  
void deleteArrs(  
 const int \*ARRAY, // Динамический массив указателей  
 const int \*ARRAY\_SORTED // Отсортированный массив указателей  
);  
  
#pragma once

**FUNCTIONS**

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* РЕАЛИЗАЦИЯ ФУНКЦИЙ \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#include "functions.h"

// Функция считывания и создания массивов

void readAndCreate(

int \*\*array, // Динамический массив указателей

int \*\*arrayFutureSourted, // Отсортированный массив указателей

int &size, // Длина массива

const char \*FILE\_INPUT // Имя выходного файла

) {

ifstream streamIn(FILE\_INPUT);

streamIn >> size; // Считывание с файла длины массива

cout << "Кол-во элементов массива из случайных чисел: " << size << endl;

(\*array) = new int[size]();

(\*arrayFutureSourted) = new int[size]();

}

// Функция наполнения массивов

void fillingOfArrays(

int \*\*array, // Динамический массив указателей

int \*\*arrayFutureSourted, // Отсортированный массив указателей

const int size // Длина массива

) {

srand(time(nullptr)); // Устанавливаем "семечко" для rand()

for (int i = 0; i < size; i++) {

(\*array)[i] = rand() % 100000; // Заполнение массива случайными значениями

(\*arrayFutureSourted)[i] = (\*array)[i];

}

}

// Функция вывода массива

void arraysOutput(

int \*\*array, // Динамический массив указателей

const int size // Длина массива

) {

for (int i = 0; i < size; i++) {

cout << "[" << i << "]\t" << (\*array)[i] << endl;

}

}

// Функция быстрой рекурсивной сортировки

void quickRecursiveSort(

int \*\*array, // Динамический массив указателей

int left, // Левая граница массива

int right // Правая граница массива

) {

//Указатели в начало и в конец массива

int L = left; // Левая граница

int R = right; // Правая граница

//Центральный элемент массива

int mid = (\*array)[(L + R) / 2];

//Делим массив

do {

//Пробегаем элементы, ищем те, которые нужно перекинуть в другую часть

//В левой части массива пропускаем(оставляем на месте) элементы, которые меньше центрального

while ((\*array)[L] < mid) {

L++;

}

//В правой части пропускаем элементы, которые больше центрального

while ((\*array)[R] > mid) {

R--;

}

//Меняем элементы местами

if (L <= R) {

int tmp = (\*array)[L];

(\*array)[L] = (\*array)[R];

(\*array)[R] = tmp;

L++;

R--;

}

} while (L < R);

if (left < R) quickRecursiveSort(array, left, R);

if (L < right) quickRecursiveSort(array, L, right);

}

// Функция линейного поиска

void sentinelLinearSearch(

int \*\*array, // Динамический массив указателей

const int SIZE, // Длина массива

int key // Ключ индексов

) {

cout << "Быстрый последовательный поиск\n";

int count\_comp = 0; // Счётчик сравнений

int last = (\*array)[SIZE - 1]; // Последний элемент массива

cout << "Искомое число: " << key << endl;

(\*array)[SIZE - 1] = key; // Число, которое мы ищем делаем последним элементом массива

int i = 0; // Переменная-счетчик

while ((\*array)[i] != key) {

i++;

count\_comp++;

}

(\*array)[SIZE - 1] = last; // Возвращаем последний элемент обратно

cout << key << " имеет индекс " << i << "\n\n";

}

// Функция смена ключа

int changeKey(

int \*\*Array, // Динамический массив указателей

int number, // Переменная для передачи значения ключа

const int SIZE // Ключ индексов

) {

int key = (\*Array)[0]; // Инициализация ключа поиска

switch (number) { // Смена ключ поиска

default:

return key;

case 1:

key = (\*Array)[SIZE / 2];

return key;

case 2:

key = (\*Array)[SIZE - 1];

return key;

}

}

// Функция поиска

void T\_find(int\*\* Array, int Size, int key) {

cout << "Искомое число: " << key << endl;

int temp = (\*Array)[Size - 1]; // Временная-переменная

(\*Array)[Size - 1] = INT32\_MAX;

int i = 0;

while ((\*Array)[i] < key && i < Size) { // Проходим последовательно весь массив

i++;

}

// Вовращаем обратно последний элемент массива

(\*Array)[Size - 1] = temp;

if ((\*Array)[i] == key) {

cout << key << " имеет индекс: " << i << "\n\n";

}

else {

cout << " Not found!\n";

}

}

// Вовращаем обратно последний элемент массива

(\*array)[SIZE - 1] = temp;

if ((\*array)[SIZE - 1] == key && !flag) {

cout << key << " имеет индекс: " << SIZE - 1 << "\n\n";

}

}

// Функция очищения памяти

void deleteArrs(

const int \*ARRAY, // Динамический массив указателей

const int \*ARRAY\_SORTED // Отсортированный массив указателей

) {

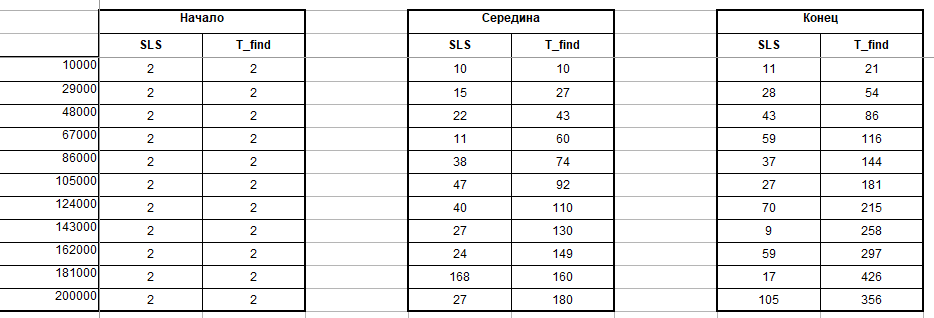
delete[] ARRAY;

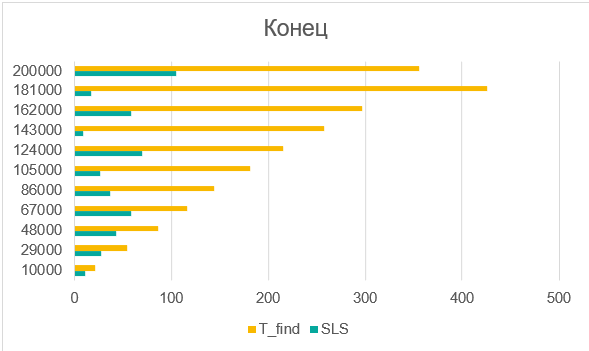
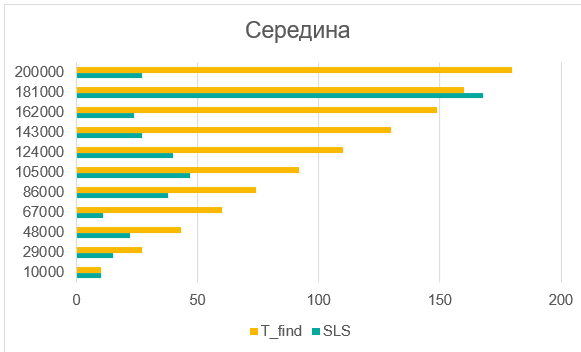
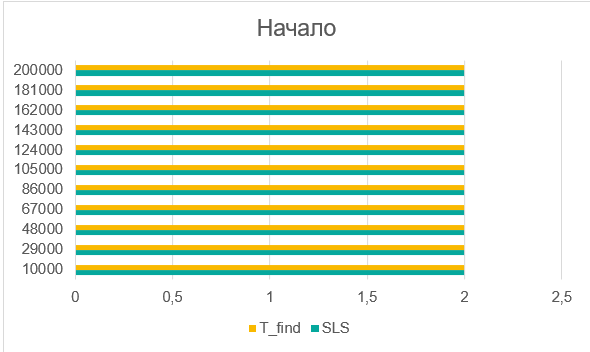
delete[] ARRAY\_SORTED;

}

# Диаграммы

Таблица соответствия времени, положения ключа и алгоритма поиска





# Вывод

В лабораторной работе были освоены принципы поиска с помощью алгоритмов быстрого последовательного поиска и последовательного поиска с фиктивной записью. Также были освоены функции для оценки длительности работы этих алгоритмов поиска и построены графики зависимостей времени от размерности данных.